L V M

A Logical Volume Manager for Linux

by

Heinz Mauelshagen

<linux.LVM@ez-darmstadt.telekom.de>

Goals

◆ Implement a flexible subsystem to handle disk storage

◆ Online allocation and relocation of storage

◆ Online extension and reduction of storage

Concept$_1$

- add an additional layer to the I/O subsystem of Linux
- gain a virtual view of physical disks or partitions
- use physical disks/partitions/ multiple devices as PVs (physical volumes)
- concatenate PVs in storage pools called VGs (volume groups)

Concept$_2$

- allocation of VG space to LVs (logical volumes) in units of PEs (physical extends)
- use LVs like disks/partitions/ multiple devices for filesystems etc.
- extend or reduce VGs and LVs online
- access VGs and LVs through device special files in /dev/VolumeGroupName/*

$Concept_3$

- configuration data called VGDA (Volume Group Descriptor Area) is stored on each PV of a VG and in work copies on filesystem
- VGDA holds all attributes of PV, VG, and LVs
- map between LEs (logical extends) of LVs and the PEs on the PVs

$Concept_4$

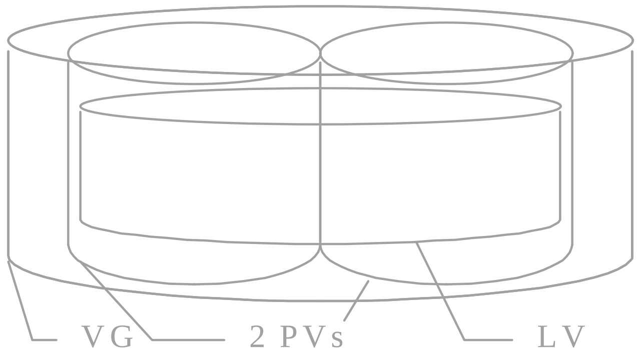
- handle the attributes and mapping information in a LVM driver/module
- add calls in "/usr/src/linux/drivers/block/ll_rw_blk.c" to call mapping function of the LVM driver/module
- create a command and a library layer

Concept$_5$

- export/import VGs to take the PVs to/from a different system
- support linear and striped (RAID0) LVs

Storage Architecture

- VG with 2 PVs and 1 LV



PV Commands

◆ pvchange	- changes attributes
◆ pvcreate	- initializes VGDA
◆ pvdata	- outputs VGDA for debugging
◆ pvdisplay	- shows PV attributes
◆ pvmove	- moves PEs between PVs
◆ pvscan	- scans periphery for PVs

VG Commands$_1$

◆ vgcfgbackup	- creates a VGDA backup
◆ vgcfgrestore	- restores a VGDA to a PV
◆ vgchange	- changes attributes
◆ vgcreate	- create a new VG
◆ vgdisplay	- shows VG attributes
◆ vgexport	- changes to “unknown”
◆ vgextend	- extends by new PV(s)

VG Commands$_2$

◆ vgimport	- changes to “known”
◆ vgmknodes	- creates device dir/nodes
◆ vgreduce	- reduces by empty PV(s)
◆ vgremove	- removes an empty VG
◆ vgrename	- renames an inactive VG
◆ vgscan	- scans periphery for VG(s)

LV Commands$_1$

◆ lvchange	- changes attributes
◆ lvcreate	- creates a new LV
◆ lvdisplay	- shows LV attributes
◆ lvextend	- extends LV in size (online!)

LV Commands$_2$

◆ lvreduce	- reduces LV in size (online!)
◆ lvremove	- removes an inactive LV
◆ lvrename	- renames an inactive LV
◆ lvscan	- scans periphery for LVs

LVM Commands

◆ lvmchange	- resets LVM (emergency)
◆ lvmdiskscan	- scans periphery for LVM usable disks (available in 0.4 alpha)

Software Metrics

- 300 hours for concept and development
- 24500 total LOC (lines of code) including all sources, headers, comments, manual pages, scripts, makefiles, README, ...
- about 21000 LOC sources and headers
- module/driver source+headers 2600 LOC
- 150 library functions in 83 modules
- 28 tools (29 including lvmdiskscan)

Example$_{1a}$

Create a VG “test” with 2 PVs (/dev/sd[kl]1) and 1 LV “tlv” containing an EXT2 filesystem:

```
# fdisk  /dev/sdk       # change the partion system id to 0xFE
# fdisk /dev/sdl        #                             “
# pvcreate /dev/sd[kl]1
pvcreate -- physical volume /dev/sdk1 successfully created
pvcreate -- physical volume /dev/sdl1 successfully created
# vgcreate test /dev/sd[kl]1
vgcreate -- INFO: using default physical extend size of 4 MB
vgcreate -- INFO: maximum logical volume size is 63.988 Gigabyte
vgcreate -- doing automatic backup of test
vgcreate -- volume group test successfully created
#
```

$\mathit{Example}_{1b}$

Now we have:

- VGDA on /dev/sd[kl]1
- character device special /dev/test/group
- VG backup in /etc/lvmconf/test.conf
- VG name in /etc/lvmtab
- VGDA work copy in /etc/lvmtab.d/test
- loaded VGDA in driver/module to access “test”

$\mathit{Example}_{1c}$

```
# lvcreate -L 300 -n tlv test
lvcreate -- doing automatic backup of test
lvcreate -- logical volume /dev/test/tlv successfully created
# mke2fs /dev/test/tlv
mke2fs 1.10, 24-Apr-97 for EXT2 FS ....
<SNIP>
Writing superblocks and filesystem accounting information: done
# mount /dev/test/tlv /usr1
....
# umount /dev/test/tlv /usr1
# vgchange -a n
```

Example$_{1d}$

Now we have:

- block device special /dev/test/tlv with capacity 300 MB
- EXT2 filesystem in /dev/test/tlv mounted on /usr1
- updated /etc/lvmtab.d/test
- /etc/lvmtab.d/test.conf renamed to /etc/lvmtab.d/test.conf.old
- new /etc/lvmtab.d/test.conf
- updated VGDA in driver/module to access /dev/test/tlv

Example$_{2a}$

Display test´s attributes normal:

```
# vgdisplay test
--- Volume group ---
VG Name                 test
VG Write Access         read/write
VG Status               available/extendable
VG #                    1
MAX LV                  31
Cur LV                  1
Open LV                 1
MAX LV Size             63.988 GB
MAX PV                  256
Cur PV                  2
Act PV                  2
VG Size                 6.184 GB
PE Size                 4 MB
Total PE                1583
Alloc PE / Size         75 / 300 MB
Free   PE / Size        1508 / 5.891 GB
```

$Example_{2b}$

Display test´s attributes verbose:

```
# vgdisplay -v test
<SNIP>
--- Logical volume ---
LV Name              /dev/test/tlv
VG NAME              test
LV Write Access      read/write
LV Status            available
LV #                 1
# open               1
LV Size              300 MB
Current LE           75
Allocated LE         75
Allocation           next free
```

... to be continued

$Example_{2c}$

```
--- Physical volumes ---
PV Name (#)          /dev/sdk1 (1)
PV Status            available / allocatable
Total PE / Free PE   1074 / 999

PV Name (#)          /dev/sdl1 (2)
PV Status            available / allocatable
Total PE / Free PE   509 / 509
```

$\textit{Example}_{3a}$

Move the LEs of /dev/test/tlv away from /dev/sdk1 to /dev/sdl1:

```
# pvmove -f /dev/sdk1    # /dev/sdl1
pvmove -- moving physical extends in active volume group test
pvmove -- doing automatic backup of test
pvmove -- 75 extends of physical volume successfully moved
#
```

$\textit{Example}_{3b}$

Reduce VG test by PV /dev/sdk1:

```
# vgreduce test /dev/sdk1
vgreduce -- doing automatic backup of test
vgreduce -- test successfully reduced
#
```

The Future

- combine the LVM with online filesystem resizing
- implement RAID1/5/10/50 in the LVM
- enhance the VGDA for additional attributes like creation and modification times
- assign UUIDs (Uniform Unique Identifiers) to VGs and PVs

Whereto

- get the LVM:
  put a “send lvm_LATEST.tar.gz” in the body of a mail to
  <ftpmail@ez-darmstadt.telekom.de>
  to get an uuencoded actual release
- ask for the LVM:
  <linux.LVM@ez-darmstadt.telekom.de>

Thank you :-)